

AD-A124 182

MAPPER: A DEBUGGING AID FOR USE WITH THE MAP
MAINTENANCE TEST PANEL(U) DEFENCE RESEARCH
ESTABLISHMENT PACIFIC VICTORIA (BRITISH COLUMBIA)
P M HOLTHAM APR 82 DREP-TM-82-4

1/1

UNCLASSIFIED

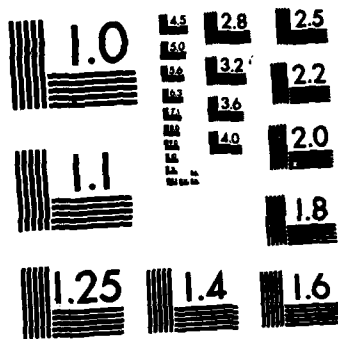
F/G 9/2

NL

END

FILMED

DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

ADA 124182



UNLIMITED
DISTRIBUTION
ILLIMITÉE

③

Victoria, B.C.

Technical Memorandum 82-4

**MAPPER: A DEBUGGING AID FOR USE WITH
THE MAP MAINTENANCE TEST PANEL**

P.M. Holtham

April 1982



DTIC FILE COPY

**Research and Development Branch
Department of National Defence**

3

DEFENCE RESEARCH ESTABLISHMENT PACIFIC
VICTORIA, B.C.

DREP Technical Memorandum 82-4

MAPPER

A DEBUGGING AID FOR USE WITH THE MAP MAINTENANCE TEST PANEL

P. M. Holtham

June 1980

Approved by:

N. Z. [Signature]

Chief/DREP



RESEARCH AND DEVELOPMENT BRANCH
DEPARTMENT OF NATIONAL DEFENCE
CANADA

ABSTRACT

A program is presented which enables communication with the Macro Arithmetic Processor manufactured by CSP Inc., through the EIA port of the Maintenance Test Panel. The user can list, edit or execute code from MAP memory, examine system flags or pseudo-memory, and with slight extension, load processors or change status words. The program is totally relocatable (including bus relocatable), and can thus be placed alongside existing software. The program was written to aid the construction and debugging of a Host Interface Controller, and was also used when making host related modifications to the Host Interface Module, or when writing the software device service routine or driver.



Accession For	
DTIC DTIC	<input checked="" type="checkbox"/>
DTIC DTIC	<input type="checkbox"/>
Unannounced	
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/of
A	Special

1. INTRODUCTION

The Macro Arithmetic Processor 200 (MAP) manufactured by CSP Inc., is a powerful 32-bit arithmetic processor than can, at least in principle, be interfaced to any main frame or minicomputer. For many brands of computer, CSP Inc. provide both the hardware and software interfacing, but in the case of our own minicomputer, a Texas Instruments 980B with the DX 980*E operating system, this support was not offered. We therefore had to develop, in house, the hardware interface, the so called Host Interface Controller or HIC, as well as the software driver and device service routine (DSR).

During this development, the Maintenance Test Panel (MTP) proved invaluable. It elevated the MAP from a 'black box', within which one could rarely, if at all, tell what was happening, to a 'front panel' unit with all the consequent benefits. These were primarily that one could enter commands directly into the MAP without need to go through the HIC, and that one could examine MAP memory to see whether attempted transfers from the host had been successful.

It soon became clear that programming the MAP through the EIA port of the MTP would greatly increase the panel's power, and allow much more rapid examination and modification of MAP memory than the conventional toggle switch entry would permit. MAPPER was therefore devised, and by using it program loops could be suitably written for the Host Interface Module (HIM) or the Central System Processing Unit (CSPU), and then executed to aid in the

location of the source of some bug. Furthermore, since the MAP could now be programmed and run entirely in the absence of a host or HIC, it was possible on several occasions to distinguish between bugs in the interface package being constructed, and periodic faults that developed in the MAP itself.

The aim of the present article, therefore, is to outline the use of MAPPER so that it might be used by others intending to interface a similar processor.

The program is clearly of major benefit for systems that are under development, since at this time the program 'LOOK', written by CSP Inc., cannot be used. It does, however, also have its role in a working system. By loading MAPPER alongside a MAP code module that is being tested and debugged (and which contains several strategically placed CSPU halt instructions), control can be passed to MAPPER and the past progress of the MAP code reviewed. After perhaps some modification of either the code or variables, the next portion of MAP code could be executed, and so on.

In the following sections, we discuss the basic commands of MAPPER together with the techniques for loading the program. Extensions of the basic commands are then discussed, and a complete assembly listing of the program is given. The nature of this article assumes the reader has some familiarity with the MAP architecture and software as covered in the CSP reference manuals.

2. BASIC COMMANDS

When ready to receive a new command, the program prompts the user with the single character '#'. Five basic commands can then be entered as follows:

#Fi	change the data bus to bus i for the A and W commands below, ie. the data will be taken from bus i.
#S	return the flag status to the flag LED display on the MTP.
#Xiabcd	start executing the code on bus i at location abcd. (Note: the data bus is left unchanged.)
#Aabcd ijkl mnop	alter the word at location abcd, replacing the old contents (ijkl) with mnop. ijkl are automatically displayed and mnop must be entered. If the command is terminated with a line-feed, the address is incremented, the next location's contents displayed, and the program waits for their modification. Alternatively, by terminating with a carriage-return, the command is finished and the program reprompts the user. A line-feed after 'ijkl' will immediately open the next location.
#Wabcd ijkl	write out the contents of abcd sequential locations starting at the address ijkl. The addresses are also printed.

Both the A and W commands process data on whichever bus has last been entered via the F command. When the program is initially loaded, the bus is set automatically to the program bus.

An incorrect character (ie. not 0-9 or A-F) entered in an argument to a command, results in the command being terminated and ignored. In the case of the A command, the contents of any opened location remain unchanged.

3 LOADING

The program is fully relocatable, and will function on any bus. To accomplish relocatability, the address switches on the panel are read to determine the program's base address. Normally, these switches will have been set during the load procedure, so caution is needed if MAPPER is entered by a 'JUMP' or 'CALL' instruction from another program. The switches must then be set accordingly.

The load procedure is as follows:

on bus 1: follow the standard EIA autoloading procedure as outlined in the MTP documentation.

on bus 2 or 3: as the MTP does not permit direct autoloading access to other than bus 1, the unit must be fooled. With MAPPER on bus 1, execute the following commands

#F2

#A0000 ???? 0880

#X20000

and the MAP will halt with bus 2 as the program bus. After pressing the address reset (avoiding the reset), one can now autoloading onto the present bus.

There is however one caution. If MAPPER resides on bus 2 or 3, and a jump to another bus is executed, then there is no means to return to MAPPER via the front panel controls.

In our system, the source program was compiled on a main frame computer using the CSP Inc. Assembler compiler. The compiled output was written to magnetic cassette which could be mounted on a Texas Instruments Silent 700 terminal for loading as outlined above.

4. EXTENSION OF BASIC COMMANDS

The five commands outlined above provide a basic set of functions for manipulating MAP core. To perform more complex functions, such as addressing pseudo-memory, changing system flags, or loading processors etc., a sequence of basic commands can be combined, or a block of code can be written into core using MAPPER, and then executed. If such procedures are used extensively, they could be incorporated into MAPPER as new commands.

Two common requirements that we had while developing the HIC and software, were to read the HIM status word HISTAT, and modify the system flag SYSFLG. These were accomplished as follows:

4.1. Addressing Pseudo-Memory

To read 16-bit words from pseudo-memory, the contents of location 3E (relative to the MAPPER start address) should be changed to 0011₁₆. This sets the data bus to 1, and will effectively add 10000₁₆ to any address supplied by the W or A commands, thereby accessing pseudo-memory. A further command #Fi will reset word 3E.

One can thus look at HISTAT by using the sequence

#Fj	set data bus to bus containing MAPPER
#A003E ???? 0011	change location 3E (relative to start address, assumed here to be zero)
#AFFCC ijkl C.R.	display contents of HISTAT
#Fi	reset word 3E.

4.2. Changing Sysflg

The HIM status word, HISTAT, could be changed by first writing into core and then executing the following code (assumed here to be at location 0800₁₆):

MOVIR	RO, \$LIT	9000 \$LIT	load up the literal (7 bits)
LLS	RO, 1	3A01	and shift left one
ADDR	RO, \$F801	9C00 F801	form the move literal instruction
MOVVM	RO, \$0807	E000 0807	and store it in next location
MOVLM	\$1FFCE	F801 FFCE	execute it to change SYSFLG
JMP	MAPPER	8000 0000	return to MAPPER (here at loc 0)

Other functions, such as loading the HIM, for example, can be achieved by similar means.

5. PROGRAM LISTING

The following pages give a listing of MAPPER obtained as output from the assembler compiler.

```

(0001) * MAPPER: MAINTENANCE TEST PANEL PROGRAM
(0002) *
(0003) *
(0004) *
(0005) * P.M. HOLTHAM, 1980, DEFENCE RESEARCH OF CANADA,
(0006) * ESQUIMALT, VICTORIA,
(0007) * B.C., CANADA.
(0008) *
(0009) *
(0010) * PROGRAM FULLY RELOCATABLE
(0011) *
(0012) *
(0013) *
(0014) * ADD OPERATORS
(0015) OPADD MOVMC, (1 .LS. 14)+(34 .LS. 8)+(X'E8')
(0016) OPADD MOVCM, (1 .LS. 14)+(30 .LS. 8)+(X'EA')
(0017) OPADD MOVDI, (1 .LS. 14)+(26 .LS. 8)+(X'OA')
(0018) OPADD MOVSO, (1 .LS. 14)+(26 .LS. 8)+(X'OD')
(0019) OPADD MOVDO, (1 .LS. 14)+(26 .LS. 8)+(X'OC')
(0020) OPADD MOVSI, (1 .LS. 14)+(26 .LS. 8)+(X'OB')
(0021) *
(0022) *
(0023) *
(0024) * DEFINE WORDS
00000E70 (0025) RETT=$0E70
00002000 (0026) NOPP=$2000
0000000A (0027) LF=$A
0000000D (0028) CR=$D
00000003 (0029) FLAGDISPLAY=3
0001FFCE (0030) SYSFLG=$1FFCE
(0031) *
(0032) *
(0033) *
(0034) * REGISTER USAGE
(0035) * R0 STACK POINTER
(0036) * R1 COUNTER IN OUT4
(0037) * R2 USED FOR I/O STATUS
(0038) * R3 ASCII I/P OR O/P CHARCCTER
(0039) * R4 PACKED HEX FOR I/P OR O/P
(0040) * R5 TEMP STORAGE - NOTE: A SUB CALL MAY CHANGE
(0041) * R6 PSEUDO BASE REG FOR RELOCATION
(0042) * R7 COUNTER IN W ROUTINE
(0043) * NOTE: ALL CALLS MUST BE MADE WITHOUT A REGISTER
(0044) * SAVE, SINCE THE STACK POINTER IS RESET
(0045) *
(0046) *
(0047) *
(0048) *
(0049) *
00000 0A80 (0050) HOME MOVDI R0,0 /SORT OUT THE RELOCATION
00001 CC0C008A (0051) MOVZM AFLAG(R6) /CLEAR ASK FLAG
00003 4060 (0052) MOVRR R6 10 /SET UP BASE REG
00004 9C000124 (0053) ADDIR R0, 10K /STACK POINTER RESET
00006 860C010A (0054) CALL R0,CALL(R6)
00008 90300023 (0055) MOVIR R3,'#'
0000A 860C008B (0056) CALL R0,WRITE(R6)

```

0000C	0AD1	(0057)	MOVDI R5,1	/CLEAR ANY INPUT CHAR
0000D	860C00A8	(0058)	CALL R0,CHAR(R6)	/WAIT FOR INPUT IN R3
0000F	92300041	(0059)	CMPIR R3,'A'	
00011	801C0061	(0060)	JMP AA(R6),EQ	
00013	92300058	(0061)	CMPIR R3,'X'	
00015	801C0028	(0062)	JMP XX(R6),EQ	
00017	92300046	(0063)	CMPIR R3,'F'	
00019	801C003A	(0064)	JMP FF(R6),EQ	
0001B	92300057	(0065)	CMPIR R3,'W'	
0001D	801C0040	(0066)	JMP WW(R6),EQ	
0001F	92300053	(0067)	CMPIR R3,'S'	
00021	801C0024	(0068)	JMP SS(R6),EQ	
00023	20C5	(0069)	HOP ABOR	
		(0070)	*	
		(0071)	*	
		(0072)	*	
00024	F051FFCE	(0073) SS	MOVMR R5,SYSFLG	/GET STATUS
00026	0DD3	(0074)	MOVSO R5,FLAGDISPLAY	
00027	2128	(0075)	HOP HOME	
		(0076)	*	
		(0077)	*	
		(0078)	*	
00028	860C00C2	(0079) XX	CALL R0,NUM(R6)	/READ BUS
0002A	3A3C	(0080)	LLS R3,12	/SHIFT TO CORRECT PLACE
0002B	4076	(0081)	MOVRR R7,R3	/STORE IT IN R7
0002C	860C0113	(0082)	CALL R0,GET4(R6)	/GET ADDRESS
0002E	EA0C0124	(0083)	MOVCM STACK(R6)	/NB: R0 MUST POINT TO
		(0084) *		/ TO STACK
00030	2601	(0085)	INCR R0,1	
00031	7050CFFC	(0086)	MOVWK R5,R0,\$CFFC	
00033	4C5E	(0087)	ADDRR R5,R7	
00034	844C0124	(0088)	MOVRL R4,STACK(R6)	
00036	E80C0124	(0089)	MOVCM STACK(R6)	/RESET THE CSW
		(0090)	EVEN	
00038	0000	(0091) ADDR2	DATA 0,0	
00039	0000			
		(0092) *		
		(0093) *		
0003A	860C00C2	(0094) FF	CALL R0,NUM(R6)	/GET BUS
0003C	3A34	(0095)	LLS R3,4	
0003D	E03C0038	(0096)	MOVRR R3,ADDR2(R6)	/STORE IT
0003F	2140	(0097)	HOP HOME	
		(0098) *		
		(0099) *		
00040	860C0113	(0100) WW	CALL R0,GET4(R6)	/GET NUMBER TO DO
00042	90502000	(0101)	MOVIR R5,NOPP	
00044	E05C0058	(0102)	MOVRR R5,INSTR(R6)	
00046	4078	(0103)	MOVRR R7,R4	/COUNT
00047	860C010A	(0104)	CALL R0,CRLF(R6)	
00049	860C0113	(0105) ENTER	CALL R0,GET4(R6)	/GET START
0004B	E04C0039	(0106)	MOVRR R4,ADDR2+1(R6)	
0004D	2004	(0107)	HOP GO	
0004E	F04C0039	(0108) LOOP2	MOVRR R4,ADDR2+1(R6)	/GET ADDRESS
00050	860C00EE	(0109)	CALL R0,OUT4(R6)	/OUTPUT IT
00052	860C0103	(0110) GO	CALL R0,SPACES(R6)	
00054	F0CC0038	(0111)	MOVRR R4,@ADDR2(R6)	/GE WORD

00056	860C00EE	(0112)	CALL R0,OUT4(R6)	/O/P IT
00058	2000	(0113)	INSTR NOP	/OTHER SUB WILL CHANGE
00059	E50C0039	(0114)	INCM ADDR2+1(R6)	/INCR ADDRESS
0005B	860C010A	(0115)	CALL R0,CRLF(R6)	
0005D	2771	(0116)	DECR R7,1	/DECR COUT
0005E	811C004E	(0117)	JMP LOOP2(R6),NE	
00060	2161	(0118)	HOP HOME	
		(0119)	*	
		(0120)	*	
		(0121)	*	
00061	90500E70	(0122)	AA MOVIR R5,RETT	
00063	E05C0058	(0123)	MOVIR R5,INSTR(R6)	
00065	860C0049	(0124)	CALL R0,ENTER(R6)	
00067	E50C008A	(0125)	INCM AFLAG(R6)	/SET THE AFLAG
00069	2002	(0126)	HOP OVER	
0006A	860C004E	(0127)	NXT CALL R0,LOOP2(R6)	
0006C	860C0103	(0128)	OVER CALL R0,SPACES(R6)	
0006E	CC0C0089	(0129)	MOVZM LFFLAG(R6)	/LINE FEED FLAG
00070	860C0113	(0130)	CALL R0,GET4(R6)	/NEW CONTENTS
00072	E20C0089	(0131)	CMPIR R5,LFFLAG(R6)	/WAS THERE A LINE FEED
00074	811C0082	(0132)	JMP INC(R6),NE	/YES. INCREMENT AND REDO
00076	E0CC0038	(0133)	MOVIR R4,@ADDR2(R6)	/STORE THEM
00078	860C00A8	(0134)	CALL R0,CHAR(R6)	/CHECK FOR TERMINATOR
0007A	9230000D	(0135)	CMPIR R3,CR	/CARRIAGE RETURN
0007C	801C0000	(0136)	JMP HOME(R6),EQ	
0007E	9230000A	(0137)	CMPIR R3,LF	/LINE FEED?
00080	811C00B7	(0138)	JMP ERROR(R6),NE	
00082	E50C0039	(0139)	INC INCM ADDR2+1(R6)	/INCR ADDRESS
00084	9030000D	(0140)	MOVIR R3,CR	
00086	860C008B	(0141)	CALL R0,WRITE(R6)	
00088	211F	(0142)	HOP NXT	
00089	0000	(0143)	LFFLAG DATA 0	
0008A	0000	(0144)	AFLAG DATA 0	
		(0145)	*	
		(0146)	*	
		(0147)	*	
0008B	4056	(0148)	WRITE MOVIR R5,R3	/OUTPUT THE CHAR
0008C	860C0093	(0149)	CALL R0,READY(R6)	/WAIT AND O/P
0008E	4E5A	(0150)	SUBRR R5,R5	/SET UP AND DO 3 NULLS
0008F	860C0093	(0151)	CALL R0,READY(R6)	/WAIT AND O/P
00091	860C0093	(0152)	CALL R0,READY(R6)	/WAIT AND O/P
00093	0BA1	(0153)	READY MOVSI R2,1	/STATUS OF PRINTER
00094	9A200001	(0154)	ANDIR R2,1	/STATUS OF PRINTER
00096	8D2C0093	(0155)	DJN R2,READY(R6)	/REWAIT IF NOT READY
00098	0CD1	(0156)	MOVDO R5,1	
00099	9230000A	(0157)	CMPIR R3,LF	
0009B	801C00A2	(0158)	JMP DLY(R6),EQ	
0009D	9230000D	(0159)	CMPIR R3,CR	
0009F	801C00A2	(0160)	JMP DLY(R6),EQ	
000A1	0E70	(0161)	BACK RETURN	
		(0162)	*	
000A2	90504A00	(0163)	DLY MOVIR R5,\$4A00	/SET UP REPEAT
000A4	1C50	(0164)	RPT R5	
000A5	90120000	(0165)	MOVIR R1,0(R1)	
000A7	2107	(0166)	HOP BACK	
		(0167)	*	

000A8	OBA1	(0168) *		
000A9	9A200030	(0169) CHAR	MOVSI R2,1	/INPUT A CHAR
000AB	9E200010	(0170)	ANDIR R2,\$30	/STATUS
000AD	803C00A8	(0171)	SUBIR R2,\$10	
000AF	811C00B7	(0172)	JMP CHAR(R6),LT	/NO CHAR YET
000B1	0AB1	(0173)	JMP ERROR(R6),NE	/TWO CHARS!
000B2	9A30007F	(0174)	MOVDI R3,1	/LEAVE PREFIC ON
000B4	860C008B	(0175)	ANDIR R3,\$7F	/STRIP UNDEFINED BITS
000B6	0E70	(0176)	CALL R0,WRITE(R6)	/ECHO
		(0177)	RETURN	
		(0178) *		
		(0179) *		
000B7	90300024	(0180) ERROR	MOVIR R3,'\$'	
000B9	860C008B	(0181)	CALL R0,WRITE(R6)	
000BB	860C008B	(0182)	CALL R0,WRITE(R6)	
000BD	860C008B	(0183)	CALL R0,WRITE(R6)	
000BF	860C008B	(0184)	CALL R0,WRITE(R6)	
000C1	21C2	(0185)	HOP HOME	
		(0186) *		
		(0187) *		
		(0188) *		
000C2	860C00A8	(0189) NUM	CALL R0,CHAR(R6)	/TO INPUT A NUMBER AND
		(0190) *		/ABORT IF ILLEGAL
000C4	92300030	(0191)	CMPIR R3,'0'	/MUST BE > OR = 0
000C6	803C00D9	(0192)	JMP ABOR1(R6),LT	/SPECIAL ABORT FOR 1ST
		(0193) *		/CHARACTER
000C8	92300039	(0194)	CMPIR R3,'9'	/MUST BE < OR = ZERO
000CA	812C00D6	(0195)	JMP OK(R6),LE	
000CC	92300041	(0196)	CMPIR R3,'A'	/MUST BE > OR = 0
000CE	803C00E9	(0197)	JMP ABOR(R6),LT	
000D0	92300046	(0198)	CMPIR R3,'F'	/MUST BE < OR = 0
000D2	802C00E9	(0199)	JMP ABOR(R6),GT	
000D4	9C300009	(0200) OK2	ADDIR R3,9	/ADD 9 FOR A-F
000D6	9A30000F	(0201) OK	ANDIR R3,\$F	/STRIP OFF GARBAGE
000D8	0E70	(0202)	RETURN	
		(0203) *		
000D9	E20C008A	(0204) ABOR1	CMPMZ AFLAG(R6)	
000DB	801C00E9	(0205)	JMP ABOR(R6),EQ	/STRAIGHT OFF TO ABOR IF
		(0206) *		/NOT IN THE 'A' ROUTINE
000DD	9230000D	(0207)	CMPIR R3,CR	
000DF	801C0000	(0208)	JMP HOME(R6),EQ	/TERMINATE THE A ROUTINE
000E1	9230000A	(0209)	CMPIR R3,LF	
000E3	811C00E9	(0210)	JMP ABOR(R6),NE	
000E5	E50C0089	(0211)	INCM LFFLAG(R6)	/SET THE LINE FEED FLAG
000E7	2702	(0212)	DECR R0,2	/DECR STACK TO GO BACK
		(0213) *		/ONE ROUTINE
000E8	0E70	(0214)	RETURN	
		(0215) *		
000E9	9030003F	(0216) ABOR	MOVIR R3,'?'	
000EB	860C008B	(0217)	CALL R0,WRITE(R6)	
000ED	21EE	(0218)	HOP HOME	
000EE	90100004	(0219) OUT4	MOVIR R1,4	/UNPACK AND OUTPUT FOUR
		(0220) *		/HEX CHARACTERS.
		(0221) *		/RESET COUNTER
000F0	3A42	(0222)	LLS R4,2	
000F1	3E4E	(0223) LOOP3	ROR R4,14	/DATA ROTATED

000F2	5038000F	(0224)	MOVKR R3,R4,\$F	/PICK OF RIGHT CHAR
		(0225) *		/AND MOVE TO R3
000F4	92300009	(0226)	CMPIR R3,9	/0-9 OR A-F
000F6	812C00FA	(0227)	JMP ALPHA(R6),LE	/JUMP FOR 0-9 ONLY
000F8	9C300007	(0228)	ADDIR R3,\$11-\$A	/A-F ADD PREFIX NOTING
		(0229) *		/ \$30 DONE NEXT LINE
000FA	9C300030	(0230)	ALPHA ADDIR R3,\$30	/0-9 ADD ASCI PREFIX
000FC	860C008B	(0231)	CALL R0,WRITE(R6)	
000FE	9E100001	(0232)	SUBIR R1,1	/DECR COUNT
00100	1810	(0233)	SKP EQ	
00101	2111	(0234)	HOP LOOP3	
00102	0E70	(0235)	RETURN	
		(0236) *		
		(0237) *		
00103	903000A0	(0238)	SPACES MOVIR R3,\$A0	/O/P 2 SPACES
00105	860C008B	(0239)	CALL R0,WRITE(R6)	
00107	860C008B	(0240)	CALL R0,WRITE(R6)	
00109	0E70	(0241)	RETURN	
		(0242) *		
		(0243) *		
0010A	9030000D	(0244)	CRLF MOVIR R3,CR	
0010C	860C008B	(0245)	CALL R0,WRITE(R6)	
0010E	9030000A	(0246)	MOVIR R3,LF	
00110	860C008B	(0247)	CALL R0,WRITE(R6)	
00112	0E70	(0248)	RETURN	
		(0249) *		
00113	860C00C2	(0250)	GET4 CALL R0,NUM(R6)	/SUBROUTINE TO READ
		(0251) *		/FOUR ASCII CHARACTERS
00115	3A3C	(0252)	LLS R3,12	/STRIP PREFIXES AND PACK
00116	4046	(0253)	MOVRR R4,R3	/R3 USED IN READ
00117	860C00C2	(0254)	CALL R0,NUM(R6)	/R4 USED FOR O/P
00119	3A38	(0255)	LLS R3,8	/R2 USED BY READ
0011A	4C46	(0256)	ADDRR R4,R3	/R1 IS ZERO SO NO
		(0257) *		/REGISTER STORE
0011B	860C00C2	(0258)	CALL R0,NUM(R6)	/R0 IS STACK POINTER
0011D	3A34	(0259)	LLS R3,4	
0011E	4C46	(0260)	ADDRR R4,R3	
0011F	860C00C2	(0261)	CALL R0,NUM(R6)	
00121	4C46	(0262)	ADDRR R4,R3	
00122	0E70	(0263)	RETURN	
		(0264) *		
		(0265) *		
00123	0800	(0266)	EVEN	
00124	0000	(0267)	STACK DATA 0	
00125		(0268)	END	

AA: 00061 (0060)(0122)
ABOR: 000E9 (0069)(0197)(0199)(0205)(0210)(0216)
ABOR1: 000D9 (0192)(0204)
ADDR2: 00038 (0091)(0096)(0106)(0108)(0111)(0114)(0133)(0139)
AFLAG: 0008A (0051)(0125)(0144)(0204)
ALPHA: 000FA (0227)(0230)
BACK: 000A1 (0161)(0166)
CHAR: 000A8 (0058)(0134)(0169)(0172)(0189)
CR: 0000D (0028)(0135)(0140)(0159)(0207)(0244)
CRLF: 0010A (0054)(0104)(0115)(0244)
DLY: 000A2 (0158)(0160)(0163)
ENTER: 00049 (0105)(0124)
ERROR: 000B7 (0138)(0173)(0180)
FF: 0003A (0064)(0094)
FLAGDISP: 00003 (0029)(0074)
GET4: 00113 (0082)(0100)(0105)(0130)(0250)
GO: 00052 (0107)(0110)
HOME: 00000 (0050)(0075)(0097)(0118)(0136)(0185)(0208)(0218)
INC: 00082 (0132)(0139)
INSTR: 00058 (0102)(0113)(0123)
LF: 0000A (0027)(0137)(0157)(0209)(0246)
LFFLAG: 00089 (0129)(0131)(0143)(0211)
LOOP2: 0004E (0108)(0117)(0127)
LOOP3: 000F1 (0223)(0234)
NOPP: 02000 (0026)(0101)
NUM: 000C2 (0079)(0094)(0189)(0250)(0254)(0258)(0261)
NXT: 0006A (0127)(0142)
OK: 000D6 (0195)(0201)
OK2: 000D4 (0200)
OUT4: 000EE (0109)(0112)(0219)
OVER: 0006C (0126)(0128)
READY: 00093 (0149)(0151)(0152)(0153)(0155)
RETT: 00E70 (0025)(0122)
SPACES: 00103 (0110)(0128)(0238)
SS: 00024 (0068)(0073)
STACK: 00124 (0053)(0083)(0088)(0089)(0267)
SYSFLG: 1FFCE (0030)(0073)
WRITE: 0008B (0056)(0141)(0148)(0176)(0181)(0182)(0183)(0184)(0217)(0231)
(0239)(0240)(0245)(0247)
WW: 00040 (0066)(0100)
XX: 00028 (0062)(0079)

END